|  |  |  |  |
| --- | --- | --- | --- |
| Subject Content | Red | Amber | Green |
| **1 Problem Solving** |
| **1.1 Algorithms** |
| 1.1.1 understand what an algorithm is, what algorithms are used for and be able to interpret algorithms (flowcharts, pseudocode, written descriptions, program code) |  |  |  |
| 1.1.2 understand how to create an algorithm to solve a particular problem, making use of programming constructs (sequence, selection, iteration) and using appropriate conventions (flowchart, pseudocode, written description, draft program code) |  |  |  |
|  |  |  |  |
| **2. Programming** |
| **2.1 Develop code** |
| 2.1.1 be able to write programs in a high-level programming language |  |  |  |
| 2.1.2 understand the benefit of producing programs that are easy to read and be able to use techniques (comments, descriptive names (variables, constants, subprograms), indentation) to improve readability and to explain how the code works |  |  |  |
| 2.1.3 be able to differentiate between types of error in programs (logic, syntax, runtime) |  |  |  |
| 2.1.5 be able to interpret error messages and identify, locate and fix errors in a program |  |  |  |
|  |  |  |  |
| **2.2 Constructs** |
| 2.2.1 understand the structural components of a program (variable and type declarations, command sequences, selection, iteration, data structures, subprograms) |  |  |  |
| 2.2.2 be able to use sequencing, selection and iteration constructs in their programs |  |  |  |
|  |  |  |  |
| **2.3 Data types and structure** |
| 2.3.1 understand the need for, and understand how to use, data types (integer, real, Boolean, char) |  |  |  |
| 2.3.2 understand the need for, and understand how to use, data structures (records, one-dimensional arrays, two-dimensional arrays |  |  |  |
| 2.3.3 understand the need for, and how to manipulate, strings |  |  |  |
| 2.3.4 understand the need for, and how to use, variables and constants |  |  |  |
|  |  |  |  |
| **2.4 Input/output** |
| 2.4.1 understand how to write code that accepts and responds appropriately to user input |  |  |  |
|  |  |  |  |
| **2.5 Operators** |
| 2.5.1 understand the purpose of, and how to use, arithmetic operators (add, subtract, divide, multiply, modulus, integer division) |  |  |  |
| 2.5.2 understand the purpose of, and how to use, relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) |  |  |  |
|  |  |  |  |
| **2.6 Subprograms** |
| 2.6.1 understand the benefits of using subprograms and be able to write code that uses user-written and pre-existing (built-in, library) subprograms |  |  |  |
|  |  |  |  |
| **3. Data** |
| **3.1 Binary** |
| 3.1.1 understand that computers use binary to represent data (numbers, text, sound, graphics) and program instructions |  |  |  |
| 3.1.2 understand how computers represent and manipulate numbers (unsigned integers, signed integers (sign and magnitude, two’s complement)) |  |  |  |
| 3.1.3 be able to convert between binary and denary whole numbers (0–255) |  |  |  |
| 3.1.4 understand how to perform binary arithmetic (add, shifts (logical and arithmetic)) and understand the concept of overflow |  |  |  |
| 3.1.5 understand why hexadecimal notation is used and be able to convert between hexadecimal and binary |  |  |  |
|  |  |  |  |
| **4. Computers** |
| **4.1 Machines and computational modelling** |
| 4.1.1 understand the input-process-output model |
|  |
| **4.2 Hardware** |
| 4.2.1 understand the function of the hardware components of a computer system (CPU, main memory, secondary storage, input and output devices) and how they work together |  |  |  |
|  |  |  |  |
| **4.3 Logic** |  |  |  |
| 4.3.1 be able to construct truth tables for a given logic statement (AND, OR, NOT) |  |  |  |
|  |  |  |  |
| **4.4 Software** |  |  |  |
| 4.4.1 know what an operating system is and how it manages files, processes, hardware and the user interface |  |  |  |
|  |  |  |  |
| **4.5 Programming Languages** |  |  |  |
| 4.5.1 understand what is meant by high-level and low-level programming languages and understand their suitability for a particular task |  |  |  |
| 4.5.2 understand what is meant by an assembler, a compiler and an interpreter when translating programming languages and know the advantages and disadvantages of each. |  |  |  |
|  |  |  |  |
| **5. Communication and the internet** |
| **5.1 Networks** |
| 5.1.1 understand why computers are connected in a network |  |  |  |
| 5.1.2 understand the different types of networks (LAN, WAN) and usage models (client-server, peer-to-peer) |  |  |  |
| 5.1.3 understand wired and wireless connectivity |  |  |  |
| 5.1.4 understand that network data speeds are measured in bits per second (Mbps, Gbps) |  |  |  |
|  |  |  |  |
| **6. The Bigger Picture** |
| **6.1 Emerging trends, issues and impact** |
| 6.1.1 understand the environmental impact of technology (health, energy use, resources) on society |  |  |  |